Processamento de Linguagem Natural

Silvio do Lago Pereira

slago@ime.usp.br

1 Introdução

Processamento de Linguagem Natural (PLN) consiste no desenvolvimento de modelos computacionais para a realização de tarefas que dependem de informações expressas em alguma língua natural (*e.g.* tradução e interpretação de textos, busca de informações em documentos e interface homem-máquina) [1,4].

Conforme [2], a pesquisa em PLN está voltada, essencialmente, a três aspectos da comunicação em língua natural:

– *som:* fonologia
– *estrutura:* morfologia e sintaxe
– *significado:* semântica e pragmática

A *fonologia* está relacionada ao reconhecimento dos sons que compõem as palavras de uma língua. A *morfologia* reconhece as palavras em termos das unidades primitivas que a compõem (*e.g. caçou* → *caç+ou*). A *sintaxe* define a estrutura de uma frase, com base na forma como as palavras se relacionam nessa frase (figura 1). A *semântica* associa significado a uma estrutura sintática, em termos dos significados das palavras que a compõem (*e.g.* à estrutura da figura 1, podemos associar o significado "*um animal perseguiu/capturou outro animal*"). Finalmente, a *pragmática* verifica se o significado associado à uma estrutura sintática é realmente o significado mais apropriado no contexto considerado (*e.g.* no contexto predador-presa, "*perseguiu/capturou*" → "*comeu*").

```
                   frase
                  /     \
                 /       \
         sujeito           predicado
         /     \          /    |    \
        /       \        /     |     \
artigo substantivo verbo artigo substantivo
   |        |        |     |        |
   o       gato    caçou   o       rato
```

Figura 1. Uma árvore sintática

Como podemos ver, PLN é uma área de pesquisa muito vasta, que envolve diversas disciplinas do conhecimento humano. Por se tratar de um assunto complexo, nesse artigo, vamos abordar apenas a análise sintática de algumas frases

em português. Mostraremos como especificar uma gramática[1] capaz de gerar um conjunto finito de sentenças e de decidir se uma determinada sentença pertence ou não à linguagem definida pela gramática. Em seguida, estenderemos essa gramática para tratar concordância de *gênero* e *número*, bem como *tempo verbal*. Finalmente, mostraremos como modificar a gramática para que ela construa a árvore sintática de uma sentença de forma automática.

2 Gramática e análise sintática

Uma *gramática* é uma especificação formal da estrutura das sentenças permitidas numa linguagem. O modo mais simples de definir uma gramática[2] é especificando um conjunto de símbolos *terminais*, denotando palavras da linguagem, um conjunto de símbolos *não-terminais*, denotando os componentes das sentenças, e um conjunto de *regras de produção*, que expandem símbolos não-terminais numa seqüência de símbolos terminais e não-terminais [3]. Além disso, a gramática deve ter um símbolo não-terminal *inicial*.

Por exemplo, a gramática a seguir define um fragmento da língua portuguesa:

Gramática 1.

$frase \Rightarrow sujeito\ predicado$
$sujeito \Rightarrow artigo\ substantivo$
$predicado \Rightarrow verbo\ artigo\ substantivo$
$artigo \Rightarrow$ `o`
$substantivo \Rightarrow$ `gato` | `rato`
$verbo \Rightarrow$ `caçou`

Nessa gramática, os símbolos terminais são `o`, `gato` e `rato`, sendo os demais símbolos não-terminais[3]. A regra de produção $frase \Rightarrow sujeito\ predicado$ estabelece que uma frase é composta de um sujeito seguido de um predicado; enquanto a regra $substantivo \Rightarrow$ `gato` | `rato` estabelece que um substantivo pode ser a palavra "*gato*" ou "*rato*". Além disso, para essa gramática, o símbolo não-terminal inicial será $frase$.

Nas gramáticas livres de contexto (do tipo que consideramos nesse artigo), o lado esquerdo de uma regra de produção será sempre um único símbolo não-terminal, enquanto o lado direito pode conter símbolos terminais e não terminais.

Como veremos a seguir, uma gramática pode ser usada tanto para *reconhecimento*, ou seja, para decidir se essa frase pertence à linguagem definida pela gramática; quanto para *geração*, ou seja, para construir uma frase pertencente à linguagem definida pela gramática.

[1] Usaremos a *gramática de cláusulas definidas* da linguagem PROLOG.
[2] Nesse artigo, tratamos apenas de gramáticas livres de contexto.
[3] Os símbolos não-terminais são escritos em *itálico*.

2.1 Reconhecimento de frases

Há duas estratégias que podem ser aplicadas para o reconhecimento de frases:

Top-down: com essa estratégia, sintetizamos a frase a ser reconhecida aplicando as regras de produção de forma progressiva, em profundidade, a partir do símbolo inicial da gramática. Por exemplo, para reconhecer a frase "*o gato caçou o rato*", procedemos da seguinte maneira:

$frase$
⇒ $sujeito$ *predicado*
⇒ *artigo substantivo predicado*
⇒ `o` *substantivo predicado*
⇒ `o gato` *predicado*
⇒ `o gato` *verbo artigo substantivo*
⇒ `o gato caçou` *artigo substantivo*
⇒ `o gato caçou o` *substantivo*
⇒ `o gato caçou o rato`

Como a frase pôde ser sintetizada, a partir das regras de produção da gramática, concluímos que ela pertence à linguagem definida pela gramática.

O processo de reconhecimento *top-down* também pode ser representado por meio de uma *árvore sintática* (gerada de forma semelhante a uma árvore de busca em profundidade, conforme a numeração indica), veja:

```
                    1-frase
                   /       \
                  /         \
          2-sujeito           5-predicado
           /     \             /    |    \
          /       \           /     |     \
    3-artigo  4-substantivo 6-verbo  7-artigo  8-substantivo
        |           |         |      |       |
        |           |         |      |       |
       "o"        "gato"    "caçou" "o"    "rato"
```

Numa árvore sintática, as folhas são sempre símbolos terminais (*i.e.* palavras da linguagem), enquanto os demais nós são sempre símbolos não-terminais (*i.e.* nomes da unidades componentes da frase).

Considere agora a frase "*o gato rato caçou o*":

$frase$
⇒ $sujeito$ *predicado*
⇒ *artigo substantivo predicado*
⇒ `o` *substantivo predicado*
⇒ `o gato` *predicado*
⇒ `o gato` *verbo artigo substantivo*

Como não há na gramática uma regra que seja capaz de derivar o símbolo terminal `rato`, a partir do símbolo não-terminal *verbo*, o reconhecimento fracassa.

Exercício 1 *Com base na gramática definida a seguir, faça o reconhecimento top-down das frases "um gato mia" e "um rato corre desesperadamente" e desenhe as árvores sintáticas.*

$frase \Rightarrow sujeito\ predicado$
$sujeito \Rightarrow$ *artigo substantivo*
predicado ⇒ *verbo_intransitivo*
predicado ⇒ *verbo_intransitivo adverbio_modal*
artigo ⇒ `um`
substantivo ⇒ `gato` | `rato`
verbo_intransitivo ⇒ `mia` | `corre`
advérbio_modal ⇒ `desesperadamente` □

Botton-up: com essa estratégia derivamos o símbolo inicial da gramática, a partir da frase a ser reconhecida, aplicando as regras de produção de forma regressiva. Por exemplo, para reconhecer a frase "*o gato caçou o rato*", fazemos:

`o gato caçou o rato`
⇒ *artigo* `gato caçou o rato`
⇒ *artigo substantivo* `caçou o rato`
⇒ $sujeito$ `caçou o rato`
⇒ $sujeito$ *verbo* `o rato`
⇒ $sujeito$ *verbo artigo* `rato`
⇒ $sujeito$ *verbo artigo substantivo*
⇒ $sujeito$ *predicado*
⇒ $frase$

Como conseguimos obter o símbolo não-terminal inicial da gramática, a partir da frase sendo reconhecida, concluímos que essa frase pertence à linguagem definida pela gramática. Vejamos um outro exemplo "*o gato rato caçou o*":

`o gato rato caçou o`
⇒ *artigo* `gato rato caçou o`
⇒ *artigo substantivo* `rato caçou o`
⇒ $sujeito$ `rato caçou o`
⇒ $sujeito$ *substantivo* `caçou o`
⇒ $sujeito$ *substantivo verbo* `o`
⇒ $sujeito$ *substantivo verbo artigo*

Como não há na gramática uma regra que seja capaz de reconhecer a seqüência *substantivo verbo artigo* como um *predicado*, o reconhecimento fracassa.

Exercício 2 *Refaça o exercício 1, usando reconhecimento botton-up.* □

2.2 Geração de frases

O processo de geração de frases funciona de forma semelhante ao reconhecimento *top-down*, exceto pelo fato que, quando um símbolo não-terminal pode ser expandido por duas ou mais regras de produção distintas, devemos criar uma ramificação para cada possibilidade. Como exemplo, vamos gerar as frases da linguagem definida pela Gramática 1.

```
                                    frase
                                      |
                               sujeito predicado
                                      |
                         artigo substantivo predicado
                                      |
                           "o" substantivo predicado
                               /                     \
      "o gato" predicado                                  "o rato" predicado
               |                                                   |
"o gato" verbo artigo substantivo               "o rato" verbo artigo substantivo
               |                                                   |
"o gato caçou" artigo substantivo               "o rato caçou" artigo substantivo
               |                                                   |
   "o gato caçou o" substantivo                    "o rato caçou o" substantivo
        /               \                                 /               \
"o gato caçou o gato"   "o gato caçou o rato"   "o rato caçou o gato"   "o rato caçou o rato"
```

Exercício 3 *Gere todas as frases da linguagem definida pela gramática a seguir:*

$frase \Rightarrow sujeito\ predicado$
$sujeito \Rightarrow pronome_pessoal \mid nome_próprio$
$predicado \Rightarrow verbo_intransitivo$
$pronome_pessoal \Rightarrow$ `você`
$nome_próprio \Rightarrow$ `Zé`
$verbo_intransitivo \Rightarrow$ `come` | `dorme` □

3 Gramática de cláusulas definidas

Prolog é a linguagem ideal para o processamento de linguagem natural [2]. Usando um recurso embutido no compilador, conhecido como notação Dcg, podemos escrever gramáticas que podem ser utilizadas tanto para o reconhecimento, quanto para a geração de frases, de forma automática. Nessa notação, as regras de produção são codificadas da seguinte forma:

Gramática 2. *Usando a notação* Dcg

```
frase --> sujeito, predicado.
sujeito --> artigo, substantivo.
predicado --> verbo, artigo, substantivo.
artigo --> [o].
substantivo --> [gato] | [rato].
verbo --> [caçou].
```

Reconhecimento automático de frases: Compilando essa gramática com o Swi-Prolog, podemos entrar no modo de consulta e digitar:

```
?- frase([o,gato,caçou,o,rato],[]).
```

A essa consulta, o sistema responderá `yes`, indicando que a frase foi reconhecida como pertencente à linguagem definida pela gramática.

Para reconhecer essa frase, o Swi-Prolog procede da seguinte maneira:

```
                              ?- frase([o,gato,caçou,o,rato],[])
                                  |
                                  | (expandindo frase)
                                  |
                  ?- sujeito([o,gato,caçou,o,rato],R0), predicado(R0,[])
                                  |
                                  | (expandindo sujeito)
                                  |
       ?- artigo([o,gato,caçou,o,rato],R1), substantivo(R1,R0), predicado(R0,[])
                                  |
                                  | (expandindo artigo: consome "o")
                                  |
          ?- substantivo([gato,caçou,o,rato],R0), predicado(R0,[])
                                  |
                                  | (expandindo substantivo: consome "gato")
                                  |
                        ?- predicado([caçou,o,rato],[])
                                  |
                                  | (expandindo predicado)
                                  |
             ?- verbo([caçou,o,rato],R2), artigo(R2,R3), substantivo(R3,[])
                                  |
                                  | (expandindo verbo: consome "caçou")
                                  |
                 ?- artigo([o,rato],R3), substantivo(R3,[])
                                  |
                                  | (expandindo artigo: consome "o")
                                  |
                         ?- substantivo([rato],[])
                                  |
                                  | (expandindo substantivo: consome "rato")
                                  |
                               sucesso
```

Observe que cada predicado tem dois parâmetros: o primeiro representa a lista de palavras a serem reconhecidas e o segundo representa a lista de palavras ainda não reconhecidas. O reconhecimento termina com sucesso se todas as palavras da frase podem ser reconhecidas (consumidas) durante a busca.

Vejamos, agora, um exemplo onde a frase não pode ser reconhecida. Digitando

```
?- frase([o,gato,rato,caçou,o],[]).
```

o sistema responderá `no`. Para decidir que essa frase não está de acordo com a gramática, o sistema procede da seguinte forma:

```
                    ?- frase([o,gato,rato,caçou,o],[])
                                |
                                | (expandindo frase)
                                |
        ?- sujeito([o,gato,rato,caçou,o],R0), predicado(R0,[])
                                |
                                | (expandindo sujeito)
                                |
?- artigo([o,gato,rato,caçou,o],R1), substantivo(R1,R0), predicado(R0,[])
                                |
                                | (expandindo artigo: consome "o")
                                |
  ?- substantivo([gato,rato,caçou,o],R0), predicado(R0,[])
                                |
                                | (expandindo substantivo: consome "gato")
                                |
            ?- predicado([rato,caçou,o],[])
                                |
                                | (expandindo predicado)
                                |
    ?- verbo([rato,caçou,o],R2), artigo(R2,R3), substantivo(R3,[])
                                |
                                | (impossível consumir "rato" como verbo)
                                |
                            fracasso
```

Note que o sistema não consegue reconhecer a frase porque, para expandir o símbolo não-terminal *verbo*, o sistema precisa consumir um verbo, que deveria ser a primeira palavra da lista. Entretanto, como a lista inicia com `rato`, e essa palavra não é um verbo, o sistema fracassa.

Exercício 4 *Usando a notação* DCG, *especifique uma gramática capaz de reconhecer as frases "o gato e o rato correm pela casa", "o gato e o rato dormem pela rua" e "o gato e o rato dormem silenciosamente".* □

Geração automática de frases: Além de reconhecer, usando o SWI-PROLOG, podemos gerar todas as frases da linguagem definida pela gramática, conforme exemplificado a seguir:

```
?- frase(F,[]).
F = [o, gato, caçou, o, gato] ;
F = [o, gato, caçou, o, rato] ;
F = [o, rato, caçou, o, gato] ;
F = [o, rato, caçou, o, rato] ;
no
```

O processo de geração é semelhante ao processo de reconhecimento; entretanto, em vez de consumir palavras durante as expansões, o sistema as produz.

Exercício 5 *Usando a notação* DCG, *codifique as gramáticas dos exercícios 1 e 3, e utilize o* SWI-PROLOG *para gerar todas as frases das linguagens definidas por essas gramáticas.* □

3.1 Concordância de gênero

Agora, vamos ampliar a gramática com artigos e substantivos femininos:

Gramática 3. *Gramática ampliada com gênero feminino*

```
frase --> sujeito, predicado.
sujeito --> artigo, substantivo.
predicado --> verbo, artigo, substantivo.
artigo --> [o] | [a].
substantivo --> [gato] | [gata] | [rato] | [rata].
verbo --> [caçou].
```

Usando a gramática para reconhecer a frase "*o gata caçou a rato*", obtemos:

```
?- frase([o,gata,caçou,a,rato],[]).
yes
```

Evidentemente, deveríamos esperar que a resposta fosse `no`. Entretanto, da forma como foi definida, a gramática não tem como garantir que artigos e substantivos concordem em gênero. Para tanto, precisamos modificar a gramática, impondo restrições de gênero:

Gramática 4. *Gramática ampliada com concordância de gêneros*

```
frase --> sujeito, predicado.
sujeito --> artigo(G), substantivo(G).
predicado --> verbo, artigo(G), substantivo(G).
artigo(m) --> [o].
artigo(f) --> [a].
substantivo(m) --> [gato] | [rato].
substantivo(f) --> [gata] | [rata].
verbo --> [caçou].
```

A modificação que fizemos consiste em definir os gêneros dos artigos e substantivos (`m` - masculino e `f` - feminino) e exigir que eles tenham o mesmo gênero `G`, quando aparecem juntos no sujeito ou no predicado de uma frase.

Para ver como a restrição de gênero funciona, analise o processo de reconhecimento do sujeito "*o gato*":

```
                    ?- sujeito([o,gato],[])
                          |
                          | (expandindo sujeito)
                          |
    ?- artigo(G,[o,gato],R0), substantivo(G,R0,[])
                          |
                          | (expandindo artigo: consome "o" e deduz G="m")
                          |
        ?- substantivo(m,[gato],[])
                          |
                          | (expandindo substantivo: consome "gato" do gênero "m")
                          |
                       sucesso
```

Agora compare com o processo de reconhecimento do sujeito "*o gata*":

```
                    ?- sujeito([o,gata],[])
                          |
                          | (expandindo sujeito)
                          |
    ?- artigo(G,[o,gata],R0), substantivo(G,R0,[])
                          |
                          | (expandindo artigo: consome "o" e deduz G="m")
                          |
        ?- substantivo(m,[gata],[])
                          |
                          | (impossível consumir "gata" do gênero "m")
                          |
                      fracasso
```

Na verdade, na linguagem definida pela Gramática 4, haverá apenas frases com artigo concordando com substantivo em gêrero:

```
?- frase(F,[]).
F = [o, gato, caçou, o, gato] ;
F = [o, gato, caçou, o, rato] ;
F = [o, gato, caçou, a, gata] ;
F = [o, gato, caçou, a, rata] ;
F = [o, rato, caçou, o, gato] ;
F = [o, rato, caçou, o, rato] ;
F = [o, rato, caçou, a, gata] ;
F = [o, rato, caçou, a, rata] ;
F = [a, gata, caçou, o, gato] ;
F = [a, gata, caçou, o, rato] ;
F = [a, gata, caçou, a, gata] ;
F = [a, gata, caçou, a, rata] ;
F = [a, rata, caçou, o, gato] ;
F = [a, rata, caçou, o, rato] ;
F = [a, rata, caçou, a, gata] ;
F = [a, rata, caçou, a, rata] ;
no
```

Exercício 6 *Modifique a Gramática 4, acrescentando também os artigos indefinidos* `um` *e* `uma`*. Em seguida, utilize o* SWI-PROLOG *para gerar todas frases dessa nova linguagem.* □

3.2 Concordância de número

Agora, vamos ampliar mais um pouco a nossa gramática, incluindo plural:

Gramática 5. *Gramática ampliada com número*

```
frase --> sujeito, predicado.
sujeito --> artigo(G), substantivo(G).
predicado --> verbo, artigo(G), substantivo(G).
artigo(m) --> [o] | [os].
artigo(f) --> [a] | [as].
substantivo(m) --> [gato] | [gatos] | [rato] | [ratos].
substantivo(f) --> [gata] | [gatas] | [rata] | [ratas].
verbo --> [caçou] | [caçaram].
```

Como podemos esperar, essa gramática não fará concordância de número:

```
?- frase([os,gato,caçou,a,rata],[]).
yes
```

Para corrigi-la, também vamos precisar impor restrições quanto ao número:

Gramática 6. *Gramática ampliada com concordância de número*

```
frase --> suj(N), pred(N).
sujeito(N) --> artigo(N,G), substantivo(N,G).
predicado(N) --> verbo(N), artigo(M,G), substantivo(M,G).
artigo(s,m) --> [o].
artigo(p,m) --> [os].
artigo(s,f) --> [a].
artigo(p,f) --> [as].
substantivo(s,m) --> [gato]  | [rato].
substantivo(p,m) --> [gatos] | [ratos].
substantivo(s,f) --> [gata]  | [rata].
substantivo(p,f) --> [gatas] | [ratas].
verbo(s) --> [caçou].
verbo(p) --> [caçaram].
```

Observe que, na regra de produção para $frase$, o número do sujeito `N` deve concordar com o número do verbo no predicado. De forma análoga, na regra para *sujeito*, artigo e substantivo devem ter o mesmo número `N`. Entretanto, na regra para *predicado*, o número do verbo `N` não precisa, necessariamente, concordar com o número `M`, do artigo e substantivo que seguem o verbo.

Veja alguns exemplos de reconhecimento com concordância de número:

```
?- frase([os,gato,caçou,a,rata],[]).
no
?- frase([os,gatos,caçou,a,rata],[]).
no
?- frase([os,gatos,caçaram,a,rata],[]).
yes
```

Exercício 7 *Modifique a Gramática 6, de modo a reconhecer sujeito composto como sendo plural. Por exemplo, para o sujeito composto "o gato e a gata" o verbo "caçar" deve estar no plural.* □

3.3 Tempo verbal

É possível ampliar ainda mais a gramática para considerar tempos verbais. Por simplicidade, vamos considerar apenas passado, presente e futuro simples.

Gramática 7. *Gramática ampliada com tempo verbal*

```
frase(T) --> sujeito(N), predicado(T,N).
sujeito(N) --> artigo(N,G), substantivo(N,G).
predicado(T,N) --> verbo(T,N), artigo(M,G), substantivo(M,G).
artigo(s,m) --> [o].
artigo(p,m) --> [os].
artigo(s,f) --> [a].
artigo(p,f) --> [as].
substantivo(s,m) --> [gato]  | [rato].
substantivo(p,m) --> [gatos] | [ratos].
substantivo(s,f) --> [gata]  | [rata].
substantivo(p,f) --> [gatas] | [ratas].
verbo(pas,s) --> [caçou].
verbo(pas,p) --> [caçaram].
verbo(pre,s) --> [caça].
verbo(pre,p) --> [caçam].
verbo(fut,s) --> [caçará].
verbo(fut,p) --> [caçarão].
```

Nessa nova versão da gramática, a cada forma do verbo "*caçar*" é associado um tempo verbal. Assim, quando gerarmos uma frase, podemos indicar o tempo verbal da frase a ser gerada. Analogamente, quando o sistema reconhecer uma frase, ele indicará o tempo verbal dessa frase.

Veja alguns exemplos de reconhecimento e geração de frases, considerando o tempo verbal:

```
?- frase(T,[o,gato,caçou,o,rato],[]).
T = pas
yes

?- frase(pre,F,[]).
F = [o,gato,caça,o,gato] <enter>
yes

?- frase(fut,F,[]).
F = [o,gato,caçará,o,gato] <enter>
yes
```

Exercício 8 *Especifique uma gramática para considerar pessoa e tempo verbal. Considere que o sujeito pode ser apenas um dos pronomes pessoais (*`eu`*,* `tu`*,* `ele`*,* `nós`*,* `vós`*,* `eles`*) e que o verbo é* `falar`*. Em seguida, utilize o* SWI-PROLOG *para gerar a conjugação desse verbo no tempo presente.* □

3.4 Construção da árvore sintática

Vamos fazer uma última modificação na nossa gramática para que ela, ao reconhecer ou gerar uma frase, construa também a sua árvore sintática.

Gramática 8. *Construindo a árvore sintática*[4]

```
frase(T,fra(S,P)) --> sujeito(N,S), pred(T,N,P).
sujeito(N,suj(A,S)) --> artigo(N,G,A), subst(N,G,S).
pred(T,N,prd(V,A,S)) --> verbo(T,N,V), artigo(M,G,A), subst(M,G,S).
artigo(s,m,art(o))  --> [o].
artigo(p,m,art(os)) --> [os].
artigo(s,f,art(a))  --> [a].
artigo(p,f,art(as)) --> [as].
subst(s,m,sub(X)) --> [X], {member(X,[gato,rato])}.
subst(p,m,sub(X)) --> [X], {member(X,[gatos,ratos])}.
subst(s,f,sub(X)) --> [X], {member(X,[gata,rata])}.
subst(p,f,sub(X)) --> [X], {member(X,[gatas,ratas])}.
verbo(pas,s,ver(caçou)) --> [caçou].
verbo(pas,p,ver(caçaram)) --> [caçaram].
verbo(pre,s,ver(caça))    --> [caça].
verbo(pre,p,ver(caçam))   --> [caçam].
verbo(fut,s,ver(caçará))  --> [caçará].
verbo(fut,p,ver(caçarão)) --> [caçarão].
```

[4] As condições entre chaves são executadas como código normal do PROLOG.

Veja um exemplo de árvore sintática construída a partir da Gramática 8:

```
?- frase(pre,A,[o,gato,caça,o,rato],[]).
A = fra(suj(art(o), sub(gato)), prd(ver(caça), art(o), sub(rato)))
Yes
```

O valor na variável `A` deve ser interpretado como a seguinte árvore sintática:

```
                fra
              /     \
            /         \
        suj              prd
        / \             / | \
       /   \           /  |  \
     art  sub      ver  art  sub
      |    |        |    |    |
      |    |        |    |    |
      o   gato    caça   o   rato
```

Exercício 9 *Modifique a gramática a seguir, de modo que ela possa ser usada para construir as árvores sintáticas das frases reconhecidas por ela.*

```
s --> sn, sv.
sn --> art, subst.
sv --> vi | vt, sn.
art --> [o].
subst --> [gato] | [rato].
vi --> [correu].
vt --> [comeu].
```

□

Referências

1. Covington, M. *NLP for Prolog Programmers*, Prentice-Hall, 1994.
2. Covington, M., Nute, D. and Vellino, A. *Prolog Programming in Depth*, Prentice-Hall, 1997.
3. Rich, E. and Knight, K. *Inteligência Artificial*, 2^a ed., Makron Books, 1995.
4. Russell, S. and Norvig, P. *Artificial Intelligence - a modern approach*, Prentice-Hall, 1995.